ROBERT A. VAN NEST (SBN 84065)
rvannest@kvn.com
CHRISTA M. ANDERSON (SBN 184325)
canderson@kvn.com
KEKER & VAN NEST LLP
633 Battery Street
San Francisco, CA 94111-1809
Telephone: (415) 391-5400
Facsimile: (415) 397-7188

SCOTT T. WEINGAERTNER (*Pro Hac Vice*)
sweingaertner@kslaw.com
ROBERT F. PERRY
rperry@kslaw.com
BRUCE W. BABER (*Pro Hac Vice*)
bbaber@kslaw.com
KING & SPALDING LLP
1185 Avenue of the Americas
New York, NY 10036-4003
Telephone: (212) 556-2100
Facsimile: (212) 556-2222

DONALD F. ZIMMER, JR. (SBN 112279)
fzimmer@kslaw.com
CHERYL A. SABNIS (SBN 224323)
csabnis@kslaw.com
KING & SPALDING LLP
101 Second Street – Suite 2300
San Francisco, CA 94105
Telephone: (415) 318-1200
Facsimile: (415) 318-1300

IAN C. BALLON (SBN 141819)
ballon@gtlaw.com
HEATHER MEEKER (SBN 172148)
meekerh@gtlaw.com
GREENBERG TRAURIG, LLP
1900 University Avenue
East Palo Alto, CA 94303
Telephone: (650) 328-8500
Facsimile: (650) 328-8508

Attorneys for Defendant
GOOGLE INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.

Plaintiff,

v.

GOOGLE INC.

Defendant.

Case No. 3:10-cv-03561-WHA

Honorable Judge William Alsup

GOOGLE'S OBJECTIONS TO ORACLE'S PROPOSED CONSTRUCTIONS

Pursuant to the Court's November 8, 2011 *Order Regarding Construction of Claims to be Tried* (Dkt. No. 603), and having met-and-conferred as directed by the Court, Google and Oracle identified three claim terms for construction in a joint submission on December 2, 2011 (Dkt. No. 637). Per the Court's Order, Google hereby states with particularity its objections to Oracle's proposed constructions.

A. The '476 Patent

Claim	Term	Google's Proposed Construction	Oracle's Proposed Construction
'476 Patent, Claim 14	computer-readable medium	any medium that participates in providing instructions to a processor for execution, including but not limited to, optical or magnetic disks, dynamic memory, coaxial cables, copper wire, fiber optics, acoustic or light waves, radio-waves and infra-red data communications	a storage device for use by a computer

This disputed construction was fully briefed in the parties' original claim construction papers. (*See* Dkt. Nos. 94, 96, 101, 102.) Google's proposed construction incorporates the ***express definition*** set forth in the '476 patent (*see* col. 5, ll. 4-41), which includes transitory signals. *See Sinorgchem Co. v. ITC*, 511 F.3d 1132, 1136 (Fed. Cir. 2007) ("the patentee must be bound by the express definition"). The Federal Circuit has held that claims to transitory signals are invalid under 35 U.S.C. § 101. *See In re Nuijten*, 500 F.3d 1346, 1353 (Fed. Cir. 2007). Adoption of Google's construction would be dispositive of the '476 patent, as this is the only asserted claim.

Oracle's proposed construction ignores the express definition in the hopes of preserving a claim that is directed to unpatentable subject matter. Although courts may interpret claims to sustain their validity in some circumstances, that maxim is applied sparingly and does not apply here. Indeed, when "other claim construction tools unambiguously resolve the claim construction dispute, considering validity would be improper." *Cross Med. Prods., Inc. v.*

Medtronic Sofamor Danek, Inc., 424 F.3d 1293, 1304 (Fed. Cir. 2005) (citing *Phillips v. AWH Corp.*, 415 F.3d 1303, 1327 (Fed. Cir. 2005) (en banc)). Because Oracle does not and cannot argue that the express definition is ambiguous, construing to preserve validity is improper. *Id.* Courts must "construe[s] claims with an eye toward giving effect to all of their terms . . . even if it renders the claims inoperable or invalid." *Haemonetics Corp. v. Baxter Healthcare Corp.*, 607 F.3d 776, 780, 781 (Fed. Cir. 2010).

Oracle's proposed construction sets aside the express definition found in the '476 patent, and makes up a new, contrived definition that gives the claim a narrower scope than the patentee sought and obtained. Oracle's desire to maintain the validity of overreaching claims is insufficient to change the proper construction. If Oracle desired to have valid claims, there are several things it could have done. Oracle might have included dependent claims having narrower scope not encompassing transitory signals; but it elected not to do so. Oracle could have sought to reissue the patent to correct the problems of claiming transitory signals; again, it elected not to do so. The Court should not now reward Oracle's strategic choices by construing the claim terms other than to have their clear intended meaning. Indeed, there is no legal precedent for doing so.

For the foregoing reasons, and as set forth more fully in Google's claim construction briefs (Dkt. Nos. 96 and 102), the Court should reject Oracle's proposed construction.

B. The '720 Patent

Claim	Term	Google's Proposed Construction	Oracle's Proposed Construction
'720 Patent, Claims 1, 10 and 19	obtain a representation or at least one class from a source definition provided as object oriented program code	load at least one class definition by compiling object oriented source code	No construction necessary. The phrase has the ordinary meaning that its constituent words give it.

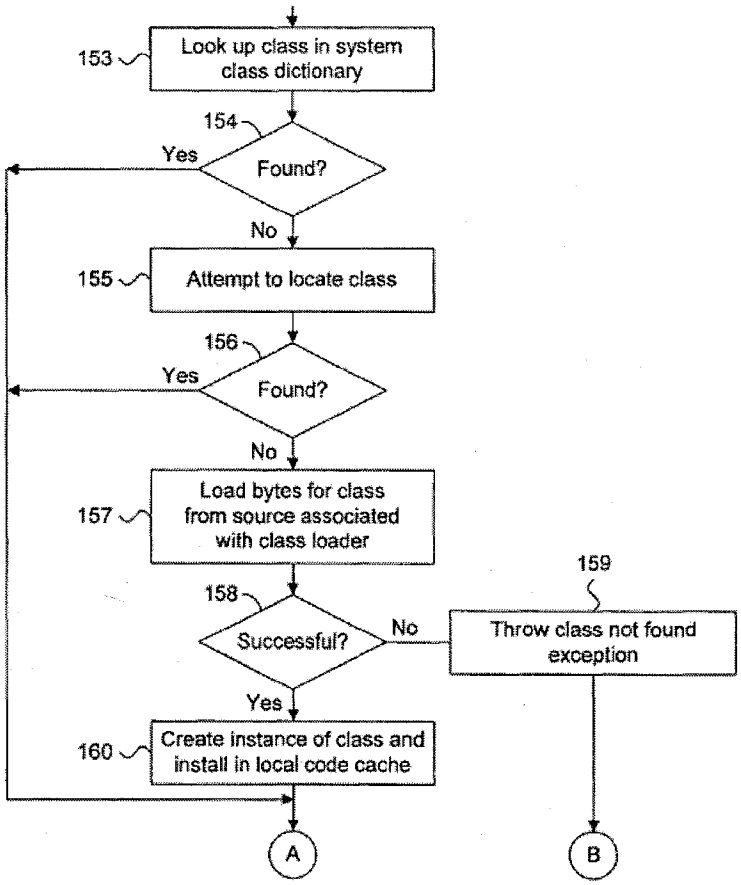
This dispute relates to the required loading by the preloader of class definitions from program ***source code*** (as opposed to the loading of class definitions from pre-compiled ***bytecode***). Construction of this term would likely be dispositive on the issue of infringement because the accused Android Dalvik virtual machine does not load a class representation from source code. Rather, source code for class files is compiled into a .dex file on the developer machine; the Android Dalvik virtual machine then loads the .dex file, which is byte code and ***not source code***.

Oracle's position that no construction is required is incorrect. A jury is not likely to understand this term without clarification. More importantly, trial is not the appropriate venue for the experts to air their differing claim interpretations; that is the province of the Court.

Here, the '720 patent specification provides strong guidance on the meaning of the claim language, which Google has incorporated into its proposed construction. Specifically, the specification describes the process for preloading classes under the heading "routine for preloading class" (column 9, lines 29–62). This description distinguishes between the most commonly employed Java class loading process, which loads class definitions from pre-compiled bytecode (*i.e.*, class files), and the special loading process required by the claims of the '720 patent, which loads class definitions from program source code.

In its May 9, 2011 Claim Construction Order, the Court described the standard Java development process where Java source files are compiled into Java class files (*i.e.*, bytecode) and then loaded into a Java virtual machine. (*See* Docket No. 137 at 3.)

The '720 patent's special loading process is distinguished from this standard process. The '720 patent illustrates the claimed invention in Figure 10, which is reproduced in relevant part here:



The related portion of the description of the "routine for preloading classes" in the '720 patent follows:

> [❶] First, the master JVM process 33 attempts to locate the class in a system class dictionary (block 153). If the class is found (block 154), no further class loading need be performed. [❷] Otherwise, the master JVM process 33 attempts to locate the class (block 155) through standard Java class path location. If the class is found (block 156), no further class loading need be performed. [❸] Otherwise, the master JVM process 33 attempts to load the bytes for the class from the ***source*** associated with the applicable bootstrap class loader 39 and system application class loader 40 (block 157). If successful (block 158), ***an instance of the class is created by compiling the source and the class instance is installed in the system class dictionary*** (block 160).

('720 patent, col. 9, ll. 42–54 (emphasis and bracketed numerals added).) The '720 patent outlines three mechanisms (numbered above) for preloading a named class into the master JVM process. First, the master JVM process checks to see if the named class has already been loaded. If not, the master JVM process attempts to locate a previously compiled class file with that class

name; this is the most frequently utilized class loading mechanism for Java programs. Failing both of those, the master JVM process utilizes a run-time compilation technique. Here, the class loader locates source code ("attempts to load the bytes for the class from the source") and then compiles that source code to create and load the class instance ("an instance of the class is created by compiling the source and the class instance is installed…"). In the context of the claimed "preloader" and the function it performs, the intrinsic record provides no other meaning of the terms "source" and "compiler."

With this foundation in mind, consider the language of independent claim 1 of the '720 patent where the third element recites: "a class preloader to obtain a representation of at least one class from a source definition provided as object-oriented program code." The words of the claim require the class representation of at least one class to be obtained from "a ***source*** definition provided as ***object-oriented program code***." The term "source" refers to source code—that is, code written by a programmer (as opposed to bytecode or native code). (*See, e.g.*, The New Hacker's Dictionary (v. 4.2.2) (Oct. 16, 1997) ("**source n.** [very common] In reference to software, 'source' is invariably shorthand for 'source code', the preferred human-readable and human-modifiable form of the program.") (available at http://www.fullbooks.com/The-New-Hacker-s-Dictionary-version-4.217.html) (last visited Dec. 9, 2011); The New IEEE Standard Dictionary of Electrical and Electronics Terms (1993) ("**source code.** Computer instructions and data definitions expressed in a form suitable for input to an assembler, compiler, or other translator. *Note:* a source program is made up of source code.").) Indeed, that is how Oracle uses the term in its own Java documentation when it distinguishes between Java source (.java files written in the Java programming language) and Java classes (.class files compiled from Java source code). (*See* The Java Tutorials, "Managing Source and Class Files, http://docs.oracle.com/javase/tutorial/java/package/managingfiles.html ("When you compile a [.java] source file, the compiler creates a different output file . . . and its extension is .class.") (last visited Dec. 9, 2011); Loading Java Classes, http://docs.oracle.com/cd/B10500_01/java.920/a96659/02_load.htm (Figure 2-1 and accompanying text describing "Java Source" as .java files and "Java Class" as .class files and stating that "Compilation of your source can be performed

. . . dynamically at runtime.") (last visited Dec. 9, 2011).) The same is true of the term "object-oriented program code." In combination there can be no doubt that "a ***source*** definition provided as ***object-oriented program code***" refers to source code. Accordingly, the claim language requires that the class preloader compile at least one class definition from source code.

The other claims of the '720 patent further support this interpretation. *Phillips*, 415 F.3d at 1314 ("Other claims of the patent in question, both asserted and unasserted, can also be valuable sources of enlightenment as to the meaning of a claim term."). For example, unasserted claim 8 requires that the source definition provided as object oriented program code must be the "Java programming language." Oracle does not and cannot dispute that the Java programming language refers to Java ***source code*** rather than Java ***bytecode***. Furthermore, while Oracle contends that the language of unasserted claim 5 demands a different result, that claim represents an anomaly that does not square with the disclosed embodiment(s). Indeed, Claims 2 and 3 distinguish between locating a "class definition"—which may be provided by a Java class file—and a "source definition" to be located if the "class definition is unavailable in the local cache." In other words, the claims themselves distinguish between source code and bytecode, and define the former as "source definition," in keeping with Google's proposed construction.

Finally, in contrast to Google's proposal, Oracle's fails to give meaning to ***all*** of the claim language. *Merck & Co. v. Teva Pharms. USA, Inc.*, 395 F.3d 1364, 1372 (Fed. Cir. 2005) ("A claim construction that gives meaning to all the terms of the claim is preferred over one that does not do so."); *see also Haemonetics Corp. v. Baxter Healthcare Corp.*, 607 F.3d 776, 780 (Fed. Cir. 2010). Standing alone, the term "class preloader" would cover a traditional Java class loader that loads class representations from Java class files. (Notably, this is the functionality Oracle accuses of infringing—the existence of a preloader that preloads class representations into an instance of a Dalvik virtual machine.) Applying the rules of claim construction, the additional language—"to obtain a representation of at least one class from a source definition provided as object-oriented program code"—must provide some additional limitation; otherwise that language is surperfluous. *Id.* In this case, that additional limitation is clear from the claim language and the specification, and requires run-time compilation of object-oriented source code

to create classes.

The disputed term is present in each of the asserted claims of the '720 patent. Because the Android Dalvik virtual machine does not include a source code compiler—and Oracle does not and cannot even make such an allegation—proper construction of this term, as proposed by Google, would preclude a finding of infringement of the '720 patent.

C. The '205 Patent

Claim	Term	Google's Proposed Construction	Oracle's Proposed Construction
'205 Patent, Claims 1 and 2	runtime	during execution of the virtual machine instructions	No construction necessary. The ordinary meaning is "during execution of the virtual machine"

Despite its claim of "no construction necessary," Oracle nevertheless proposes a construction that is contrary to the ordinary meaning of the term "runtime" and inconsistent with the claim language. This term requires construction because the parties dispute whether runtime occurs during execution of the virtual machine ***instructions***, or during the execution of the virtual machine. Oracle's omission of the term "instructions" improperly expands the concept of "runtime," and should be rejected.

The Federal Circuit has held that the language of the claim itself is "highly instructive" of the meaning of a disputed claim term. *Phillips*, 415 F.3d at 1314 ("[T]he context in which a term is used in the asserted claim can be highly instructive" as to the meaning of the term.) Here, the preamble of claim 1 of the '205 patent indicates that the claim is directed to "a method for increasing the ***execution*** speed ***of virtual machine instructions*** at runtime." ('205 patent at col. 13, ll. 43–44 (emphasis added).) Thus, the claim language itself is explicit that the entire context of the term "runtime" relates to the execution of virtual machine ***instructions***—and not just the execution of a virtual machine in general. Indeed, it would be illogical to expand the term "runtime" to include execution of the virtual machine, when virtual machine instructions are ***not*** being executed. In such a scenario there would be no reason or ability to increase the "execution

speed of virtual machine instructions"—since virtual machine instructions are not being executed—and thus no need for the claimed invention at all.

Along those same lines, claim 1 of the '205 patent is replete with references to virtual machine instructions, without reciting a "virtual machine" (alone) as a limitation. As a result, Oracle's proposed construction of "during the execution of ***the virtual machine***" finds no support in the remaining language of the claim and begs the question "what virtual machine?" This would not be helpful to the jury. In contrast, Google's proposed construction is consistent with the claim's repeated use of "virtual machine instructions" and makes it clear that "runtime" is during the execution of the virtual machine instructions that are referenced throughout the claim.

Oracle's proposed construction of the term is also inconsistent with the intrinsic evidence in the '205 patent specification. Although the term "runtime" is not found anywhere in the specification, the specification's use of the word "run" sheds light on the meaning of "runtime." The '205 patent specification uses the term "run" four times, and every use supports Google's position that "runtime" refers to execution of virtual machine instructions. For example, the specification discloses that:

> An advantage of utilizing virtual machine instructions is the flexibility that is achieved since ***the virtual machine instructions may be run***, unmodified, on any computer system that has a virtual machine implementation, making for a truly portable language.

Id. at 1:47-51 (emphasis added). This passage suggests that "runtime" relates to the time when virtual machine ***instructions*** are run, which is consistent with Google's proposed construction of "during execution of the virtual machine instructions." The remaining uses of the term "run" also suggest that "runtime" relates to execution of instructions, since these passages disclose running a "computer program," a "program," and "native code," each of which are made up of instructions. (*See* '205 patent at 1:17-21 ("porting an existing ***computer program*** to run on a different computer platform"), 9:55-57 ("the faster ***the program*** will run"), 2:5-7 ("***native code*** . . . does not always run faster than code executed by an interpreter") (emphasis added).

Accordingly, this Court should not adopt Oracle's proposed "ordinary meaning" construction of this term. Notably, Oracle has advanced two different infringement theories with respect to the '205 patent, accusing both the dexopt program ("dexopt theory") and the Dalvik Just-in-Time compiler ("JIT theory") of infringement. Adoption of Google's proposed construction is likely to be dispositive with respect to at least the dexopt theory because the dexopt program does not "generate . . . a new virtual machine instruction" at runtime – *i.e.*, during execution of the virtual machine instructions.

DATED: December 9, 2011

KEKER & VAN NEST, LLP

By: /s/ Christa M. Anderson

ROBERT A. VAN NEST (SBN 84065)
rvannest@kvn.com
CHRISTA M. ANDERSON (SBN 184325)
canderson@kvn.com
KEKER & VAN NEST LLP
633 Battery Street
San Francisco, CA 94111-1809
Telephone: (415) 391-5400
Facsimile: (415) 397-7188

SCOTT T. WEINGAERTNER (*Pro Hac Vice*)
sweingaertner@kslaw.com
ROBERT F. PERRY
rperry@kslaw.com
BRUCE W. BABER *(Pro Hac Vice)*
bbaber@kslaw.com
KING & SPALDING LLP
1185 Avenue of the Americas
New York, NY 10036-4003
Telephone: (212) 556-2100
Facsimile: (212) 556-2222

DONALD F. ZIMMER, JR. (SBN 112279)
fzimmer@kslaw.com
CHERYL A. SABNIS (SBN 224323)
csabnis@kslaw.com
KING & SPALDING LLP
101 Second Street – Suite 2300
San Francisco, CA 94105
Telephone: (415) 318-1200
Facsimile: (415) 318-1300

IAN C. BALLON (SBN 141819)
ballon@gtlaw.com
HEATHER MEEKER (SBN 172148)
meekerh@gtlaw.com
GREENBERG TRAURIG, LLP
1900 University Avenue
East Palo Alto, CA 94303
Telephone: (650) 328-8500
Facsimile: (650) 328-8508

ATTORNEYS FOR DEFENDANT
GOOGLE INC.